AMENDMENT TO THE SPECIFICATION

Please amend the specification as follows.

Please replace paragraph [36] on page 7 with the following:

-- Figure 2 shows a sequence diagram of the process of reading ahead used in the system of Figure 1. In steps 1a-1c, to retrieve a web page (i.e., HTML page) from web server 109, the web browser 103 on PC 101 issues an HTTP GET request. It is observed that the HTTP protocol also supports a GET IF MODIFIED SINCE request wherein a web server (or a proxy server) either responds with a status code indicating that the URL has not changed or with the URL content if the URL has changed since the requested date and time. For the purposes of explanation, the HTML page is addressed as URL "HTML." When the GET request is received, the downstream server 105 checks its cache 115 to determine whether the requested URL has been previously visited. If the downstream proxy server 105 does not have URL HTML stored in cache 115, the server 105 relays this request, GET URL "HTML", to upstream server ~~117~~ 107. --

Please replace paragraph [37] on pages 7 and 8 with the following:

-- The upstream server ~~117~~ 107 in turn searches for the URL HTML in its cache 117; if the HTML page is not found in cache 117, the server ~~117~~ 107 issues the GET URL HTML request to the web server 109 for the HTML page. Next, in steps 2a-2c, the web server 109 transmits the requested HTML page to the upstream server ~~117~~ 107, which stores the received HTML page in cache 117. The upstream server ~~117~~ 107 forwards the HTML page to the downstream server 105, and ultimately to the web browser 103. The HTML page is stored in cache 115 of the downstream server 105 as well as the web browser's cache (not shown). In step 3, the upstream server 107 parses the HTML page and requests the embedded objects within the HTML page from the web server 109; the embedded objects are requested prior to receiving corresponding embedded object requests initiated by the web browser 103. Although Figure 2 shows steps 2a-2c, and 3 in sequence, the upstream server 107 can perform steps 2b and 3 in parallel. --

Please replace paragraph [64] on pages 15 and 16 with the following:

-- Figure 7 is a sequence diagram of the process of reading ahead used in the system of Figure 6. In step 1, the web browser ~~101~~ 103 sends a GET request (e.g., GET x.html) to the downstream server 601. The downstream server 601 checks the URL object cache 603 (step 2) to determine whether x.html is stored in the URL object cache 603; if the content is stored in cache 603, the downstream server 601 forwards the content to the browser 103. Otherwise, the downstream server 601 writes the request in the Outstanding Request table 605 and sends the GET request to the upstream server 607 (step 3). In this case, the web browser 103 and the downstream server 601 have not encountered the requested HTML page before. However, in the event that the web browser 103 has requested this HTML page in the past or the downstream server 601 has stored this HTML previously, the latest time stamp is passed to the upstream server as a conditional GET request (e.g., GET IF MODIFIED SINCE 9/22/00). In this manner, only content that is more updated than the time stamp are retrieved. In step 4, the upstream server 607 checks the URL object cache 611 in response to the received GET x.html request. Assuming x.html is not found in the URL object cache 611, the upstream server 607 forwards the GET x.html request to the web server 109, per step 5. Accordingly, the web server 109, as in step 6, returns the web page to the upstream server 607. In turn, the upstream server 607 forwards the web page to the downstream server 601, as in step 7, and stores the web page in the URL object cache 611, per step 8 (if the web page is cacheable). Prior to forwarding the web page to the downstream server 601, the upstream server 607 parses the web page to determine this list of embedded objects that it will read ahead, based upon the read-ahead criteria that were discussed with respect to Figure 2. An Expected Objects List is then attached to the web page when it is forwarded and the list is stored in the Unsolicited URL table 613. In step 9, the downstream server 601 removes the attached "Expected These Objects" list and sends the received web page to the web browser 103. At this time, the downstream server 601 deletes the corresponding entry in the Outstanding Request table 605, stores the received web page in the URL object cache ~~611~~ 603 (if the web page is cacheable) and stores the list of expected objects in the Expected URL Objects table 615 (step 10). --

Please replace paragraph [65] on page 16 with the following:

-- Concurrent with steps 7 and 8, the upstream server 607 requests ("reads ahead") the embedded objects of the web page using a series of GET embedded object requests (step 11). In step 12, the web server 109 returns the embedded objects to the upstream server 607. The upstream server 607 forwards the embedded objects to the downstream server 601 based on a forwarding criteria (as previously discussed with respect to Figure 2), removes the embedded object's entry from the Unsolicited URL table ~~611~~ 613 and also stores these embedded objects in the URL object cache ~~611~~ 613 (if they are cacheable) (step 13). If the embedded object arrives at downstream server 601 prior to a request to for the object arriving from web browser 103, downstream server 601 will store the object in the URL object cache 603 and remove the entry for the object from the Expected URL Object table 615. Downstream server 601 will store all of the embedded objects in the URL object cache 603, even if they are not cacheable, in order to save them for when web browser 103 requests them. Uncacheable objects placed in the cache 603 for this purpose are removed from the cache 603 once they have been sent to web browser 103. In the case of Figure 7, however, the embedded object arrives at downstream server 601 after the web browser 103 has requested it, as described below. --

Please replace paragraph [68] on page 17 with the following:

-- Figure 8 is a diagram of a computer system that can be configured as a proxy server, in accordance with an embodiment of the present invention. Computer system 801 includes a bus 803 or other communication mechanism for communicating information, and a processor 805 coupled with bus 803 for processing the information. Computer system 801 also includes a main memory 807, such as a random access memory (RAM) or other dynamic storage device, coupled to bus 803 for storing information and instructions to be executed by processor 805. In addition, main memory 807 may be used for storing temporary variables or other intermediate information during execution of instructions to be executed by processor 805. Computer system 801 further includes a read only memory (ROM) 809 or other static storage device coupled to bus 803 for storing static information and instructions for processor 805. A storage device 811, such as a magnetic disk or optical disk, is provided and coupled to bus 803 for storing information and instructions. For example, the storage device ~~711~~ 811 (e.g., disk drive, hard drive, etc.) may store the tables utilized by the proxy servers 601 and 607 of the system of Figure 6. --